Escola de Matemática Aplicada

Aula Prática 4 - ALN - Mínimos Quadrados

Aluno: *Gustavo Murilo Cavalcante Carvalho*

E-mail: *gustavomurilo012@gmail.com*

Programa: *Bacharelado em Matemática Aplicada*

Disciplina: *Álgebra Linear Numérica*

Professor: *Antonio Carlos Saraiva Branco*

Data: *10 de junho de 2024*

Sumário

Problema 1 - Cobb-Douglas

1 - a)

A seguir está um modelo formulado por Charles Cobb e Paul Douglas, que estabelece a produção (P) em função do capital investido (L) e do trabalho (K). Esses valores são todos maiores ou iguais a 0.

Sejam $\alpha, b \in \mathbb{R}$, a função g: $(\mathbb{R}_+)^2 \to \mathbb{R}$. Considere a notação $P = g(L, K)$.

$$P = bL^{\alpha}K^{1-\alpha}$$

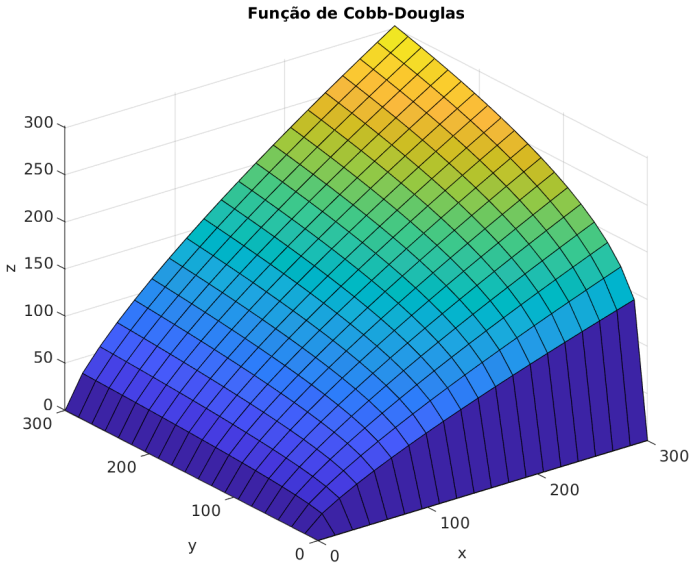
Com base em um conjunto de dados, queremos obter os parâmetros α e b que melhor se encaixam com esses dados. Para isso, é preciso fazer uma regressão exponencial.

Observação: A fim de ser mais conciso, chamarei o Método dos Mínimos Quadrados, utilizado para resolver problemas de otimização, pela sua sigla MMQ.

Abordagem inicial (1) - MMQ sobre um plano

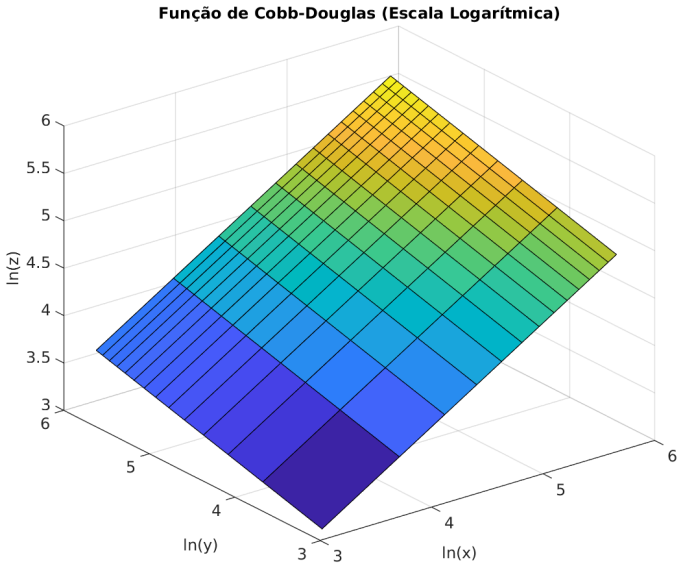
Um técnica para isso, é trabalhar com o conjuntos de dados em escala logarítmica, tornando a superfície que é a imagem da função em um plano. Assim, reduzimos a regressão exponencial a um simples problema de regressão linear sobre um novo conjunto de dados.

Definida a função, vamos pegar uma restrição dela no nosso conjunto de dados. Sejam L_1, K_1, $P_1 \in (\mathbb{R})^{24}$. Temos que $P_1 = P|_{L_1 \times K_1}$.



$g(L, K)$ com $\alpha = 0.74461$, $b = 1.0071$

$P = bL^{\alpha}K^{(1-\alpha)}$



$\ln(g(L, K))$ com $\alpha = 0.74461$, $b = 1.0071$

$\ln(P) = \ln(b) + \alpha \ln(L) + [1 - \alpha] \ln(K)$

Visto o efeito geométricos dessa mudança de escala, vamos ver a o efeito algébrico, o porquê dessas grandezas passarem a se relacionar de forma linear:

$$\ln(P) = \ln(bL^{\alpha}K^{1-\alpha})$$

$$\ln(P) = \ln(b) + \alpha\ln(L) + [1-\alpha]\ln(K)$$

Essa é a parametrização de um plano em $\left(\mathbb{R}_{+}^{*}\right)^{3}$.Também note que os coeficientes de 2 direções estão relacionados, ambos dependem de α. Essa parametrização tem um propriedade interessante, ela possui um ponto fixo p, que pertence ao plano independente da escolha de α.

Sobre isso, não é possível, utilizando o MMQ, garantir que os coeficientes encontrados satisfaçam essa condição. Então o modelo obtido pode não ser a função de Cobb-Douglas.

Aplicamos o logaritmo ao nosso conjunto de dados, obtendo os vetores $P_{\ln},\ L_{\ln},\ K_{\ln} \in \mathbb{R}^{24}$.

Ignoramos a restrição (relação entre as variáveis) e fazemos uma regressão linear múltipla, obtendo os coeficientes para um plano em $\mathbb{R}^3$, que restringimos a $\left(\mathbb{R}_{+}\right)^{3}$. Para isso criamos a matriz

$$A = \begin{bmatrix} 1 & | & | \\ \vdots & L_{\ln} & K_{\ln} \\ 1 & | & | \end{bmatrix}$$

Então utilizamos o MMQ para obter os coeficientes que descrevem a superfície que mais se aproxima do conjunto de dados. O que consiste em resolver para $\bar{x}$ a seguinte equação

$$A^T A\bar{x} = A^T P_{\ln}$$

Disso, obtemos o seguinte o plano

$$P_{\ln} = -0.069 + 0.769\ L_{\ln} + 0.247\ K_{\ln}$$

Então retornamos para escala anterior, enfim obtendo a função que melhor aproxima os dados. Então elevamos e a cada lado da equação, obtendo

$$P = 0.933\ L^{0.769}\ K^{0.247}$$

O que não é a função de Cobb-Douglas como definida no problema, dado que

$$0.769 + 0.247 = 1.016 \Rightarrow 0.247 \neq 1 - 0.769$$

.

Abordagem principal (2) - MMQ sobre uma reta

Vide a inadequação da abordagem anterior, iremos resolver o problema de outra forma. Retomamos o problema a partir da seguinte equação:

$$\ln(P) = \ln(b) + \alpha\ln(L) + [1-\alpha]\ln(K)$$

Fazendo algumas manipulações algébricas, conseguimos relacionar esses dados sobre uma reta. O que torna possível a obtenção destes parâmetros com uma regressão linear simples.

$$[\ln(P) - \ln(K)] = \ln(b) + \alpha\ [\ln(L) - \ln(K)]$$

$$\underbrace{\ln(P/K)}_{y} = \ln(b) + \alpha\ \underbrace{\ln(L/K)}_{x}$$

Sejam $y, x \in \mathbb{R}^{24}$, $B \in \mathbb{R}^{24\times 2}$. Convém reescrever a equação em forma matricial

$$y = \begin{bmatrix} 1 & | \\ \vdots & x \\ 1 & | \end{bmatrix} \begin{bmatrix} \ln(b) \\ \alpha \end{bmatrix} = B \begin{bmatrix} \ln(b) \\ \alpha \end{bmatrix} = B\,c$$

Então, para achar α e $\ln(b)$, os coeficientes da reta que melhor se encaixa aos pontos, usamos o MMQ, o que consiste em resolver para $\bar{c}$ a seguinte equação:

$$B^T\,B\,\bar{c} = B^T\,y$$

Fazendo isso, obtemos os parâmetros da função de Cobb-Douglas

$$\bar{c} = \begin{bmatrix} 0.007044 \\ 0.74461 \end{bmatrix} = \begin{bmatrix} \ln(b) \\ \alpha \end{bmatrix}$$

$$\Rightarrow\ a = 0.74461\,, \quad e^{\ln(b)} = b = 1.0071$$

Valendo que

$$P = 1.0071\ L^{0.74461}\ P^{0.25539}$$

$$P \approxeq 1.007\ L^{0.745}\ P^{0.255}$$

.

1 - b)

No item anterior, concluí que a segunda abordagem é a mais adequada porque o modelo obtido nela é a função de Cobb-Douglas. Já na primeira abordagem, obtemos uma função que aproxima bem a função de Cobb-Douglas, apresentando diferenças significativas apenas para valores maiores de L e K.

O resultado dos dois modelos é muito parecido. No intervalo que tomamos L e K, ambas as funções aproximam bem os dados reais. Inclusive, a soma dos erros ao quadrado (SEQ) dos dois modelos é bem parecida.

Ano	P (Real)	P (Abordagem 1)	P (Abordagem 2)
1910	159	161,86	161,76
1920	231	236,49	236,07

	Abordagem 1	Abordagem 2
SEQ	2702.8	2670.7

1 - Implementação no MATLAB

Obtenção dos dados

```
>> Data = readmatrix("Datasets/cobb_douglas.csv");

% Vetores com a amostragem
>>  Anos = Data(:,1); P = Data(:,2); L = Data(:,3); K = Data(:,4);

% Transformação de escala
>> P_ln = log(P); L_ln = log(L); K_ln = log(K);

n = size(P, 1);
```

Abordagem_1.m – Script

```
A = [ones(n, 1), L_ln, K_ln];

x = Gaussian_Elimination_4((A' * A), (A' * P_ln));
% x = [-0.069214,   0.76887,    0.24711]

b_1 = exp(x(1)); % b = 0.93313

SEQ_1 = sum( (P - b_1 .* L.^x(2) .* K.^x(3) ) .^ 2 );
% SEQ_1 = 2702.8

resultado_1 = [Anos, P, (b_1 .* L.^x(2) .* K.^x(3))];

resultado_1;
% Ano          P     Abordagem_1
% 1899       100       100.44
% 1900       101       106.03
% 1901       112       111.63
% 1902       122       119.03
% 1903       124        125.1
% 1904       122       125.93
% 1905       143       131.58
% 1906       152       141.92
% 1907       151        149.6
% 1908       126        137.1
% 1909       155       156.54
% 1910       159       161.86 <----
% 1911       153       164.23
% 1912       177       172.08
% 1913       184        174.8
% 1914       169       172.76
% 1915       189       180.04
```

```
% 1916       225       209.35
% 1917       227       228.95
% 1918       223       236.73
% 1919       218       235.41
% 1920       231       236.49 <----
% 1921       179       191.21
% 1922       240       207.83
```

Abordagem_2.m – Script

```
y_2 = log(P./K); x_2 = log(L./K);
B = [ones(length(P), 1), x_2];

c = Gaussian_Elimination_4((B' * B), (B' * y_2));

alpha = c(2); % alpha = 0.74461

b_2 = exp(c(1)); % b = 1.0071

SEQ_2 = sum((P - b_2 .* L.^(alpha) .* K.^(1 - alpha)).^2);
% SEQ_2 = 2670.7

resultado_2 = [Anos, P, (b_2 .* L.^(alpha) .* K.^(1 - alpha))];
% Ano          P      Abordagem_2
% 1899       100       100.71
% 1900       101       106.25
% 1901       112       111.79
% 1902       122       119.09
% 1903       124       125.12
% 1904       122       126.02
% 1905       143       131.66
% 1906       152       141.87
% 1907       151       149.48
% 1908       126       137.48
% 1909       155       156.49
% 1910       159       161.76 <----
% 1911       153       164.16
% 1912       177       171.88
% 1913       184       174.62
% 1914       169       172.74
% 1915       189       180.04
% 1916       225       208.73
% 1917       227       228.06
% 1918       223        235.9
% 1919       218       234.84
% 1920       231       236.07 <----
% 1921       179       192.23
% 1922       240        208.5
```

Problema 2

Modelagaem

Para essa questão, temos duas bases de dados, uma para o treinamento de um modelo, uma função que identifica se pacientes tem câncer.

Considerando a base de dados para treinamento, construímos com suas 10 primeiras uma matriz $D_1 \in \mathbb{R}^{280\times 10}$, onde cada linha representa um paciente e cada dimensão representa uma característica supostamente relevante para a identificação de câncer de mama. A última coluna da base de dados, representada por $b_1 \in \mathbb{R}^{280}$, contém em cada linha a informação sobre a presença (1) ou ausência (-1) de câncer no paciente da linha correspondente em D_1.

Queremos um modelo que com base nos valores das 10 características supracitadas retorne se o paciente tem ou não câncer. Para essa classificação, definimos um hiperplano H que divide o nosso espaço ($\mathbb{R}^{10}$) em dois. De modo que, os pontos abaixo dele representam os pacientes saudáveis, enquanto os pontos que coincidem ou estão acima de H representam os pacientes que têm câncer.

$$H : \alpha_0 + \sum_{i=1}^{10} \alpha_i \, x_i = 0$$

Queremos então achar o conjunto de constantes que tragam o melhor encaixe do hiperplano ao conjunto de dados (os 280 pontos em $\mathbb{R}^{10}$). Isto é, queremos fazer uma regressão linear. Para isso, construímos a seguinte matriz

$$M_1 = \begin{bmatrix} 1 & | \\ \vdots & D_1 \\ 1 & | \end{bmatrix}$$

Dispondo dessa matriz, usamos o MMQ para descobrir o vetor $\bar{y_1}$ que contém as constantes desejadas. Assim, basta resolver a equação

$$(M_1)^T \, M_1 \, \bar{y_1} = (M_1)^T \, b$$

No MATLAB, uso a função “Gaussian_Elimination_4” que retorna o seguinte vetor:

$$\bar{y} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \\ \alpha_9 \\ \alpha_{10} \end{bmatrix} = \begin{bmatrix} -6.21 \\ 15.90 \\ 1.55 \\ -5.07 \\ -7.18 \\ 1.27 \\ -0.92 \\ 0.52 \\ 1.95 \\ -0.04 \\ 0.77 \end{bmatrix}$$

Quanto à base de dados de teste, construímos de forma análoga as matrizes notadas por D_2 e b_2 e M_2.

Treinado o modelo, é esperado que ele classifique bem os dados do treinamento (é claro) assim como outra amostras, o que indica que o modelo possui capacidade de generalização.

Para classificar os pacientes em cada base de dados, basta multiplicar a matriz (M_1 ou M_2) pelo vetor $\bar{y}$. Fazemos isso obtendo, $p_1, p_2 \in \mathbb{R}^{240}$, que são previsões de câncer em pacientes.

Matriz de Confusão

Uma matriz de confusão é, essencialmente, uma matriz que contém a quantidade de previsões acetadas, positivo verdadeiro (P_V) e negativo verdadeiro (N_V), e quantidade de previsões erradas, falso positivo (P_F) e falso negativo (N_F). Essas informações são dispostas na forma:

P_V	N_F
P_F	N_V

Dela, obtemos algumas métricas:

Acurácia = $\frac{P_V+N_V}{P_V+P_F+N_F+N_V}$ Recall = $\frac{P_V}{P_V+N_F}$ Omissão de Alarme = $\frac{N_F}{P_V+N_F}$

Precisão = $\frac{P_V}{P_V+P_F}$ Falso Alarme = $\frac{P_F}{P_F+N_V}$

Dos exemplos, obtivemos os seguintes dados:

Treinamento

$$C_1 = \begin{array}{|c|c|}\hline 106 & 14 \\ \hline 8 & 152 \\ \hline \end{array}$$

Acurácia = 0.921 Recall = 0.883 Omissão de Alarme = 0.116

Precisão = 0.929 Falso Alarme = 0.05

Teste

$$C_2 = \begin{array}{|c|c|}\hline 80 & 6 \\ \hline 25 & 169 \\ \hline \end{array}$$

Acurácia = 0.889 Recall = 0.930 Omissão de Alarme = 0.069

Precisão = 0.761 Falso Alarme = 0.128

Como esperado, o modelo é, no geral, mais preciso ao classificar os dados que usou para o seu treino. Ainda assim, vemos números relativamente pequenos de erros no teste, o que indica que o modelo tem uma boa capacidade de generalização.

No exemplo, o exemplo classificou menos falsos negativos sobre a base de teste. Não podemos tirar nenhuma conclusão desse acontecido. Contudo, vale ressaltar que é no mínimo uma feliz coincidência que os piores casos, aqueles que a pessoa tem a doença e não é notificada para iniciar o tratamento, são os que menos acontecem.

2 - Implementação no MATLAB

Matriz_Confusao.m – Função

```
function [C, acuracia, precisao, recall, alarme_falso, omissao] =
Matriz_Confusao(prev ,info)

  PV = dot((info >= 0), (prev >= 0));  % Positivo Verdadeiro
  NF = dot((info >= 0), (prev < 0));   % Falso Negativo
  PF = dot((info < 0), (prev >= 0));   % Falso Positivo
  NV = dot((info < 0), (prev < 0));    % Negativo Verdadeiro

  C = [PV, NF; PF, NV];

  acuracia = (PV + NV) / (PV + NF + PF + NV);
  precisao =        PV / (PV + PF);
  recall =          PV / (PV + NF);
  alarme_falso =    PF / (PF + NV);
  omissao =         NF / (NF + PV);
end
```

Questao_2.m – Script

```
Data_cancer_train = readmatrix("cancer_train_2024.csv");
Data_cancer_test = readmatrix("cancer_test_2024.csv");

samples_train = Data_cancer_train(:, 1:10);
sign_train = Data_cancer_train(:, 11);
samples_test = Data_cancer_test(:, 1:10);
sign_test = Data_cancer_test(:, 11);

% Obtem as dimensoes da matriz com as amostras
n = size(samples_train, 1);
m = size(samples_train, 2);

A = [ones(n, 1), samples_train];
% Obtem, usando minimos quadrados, os parametros para regressao linear
y = Gaussian_Elimination_4((A' * A), (A' * sign_train));
```

```
prev_1 = A * y;

B = [ones(n, 1), samples_test];
% Usamos o modelo para classificar os dados do teste
prev_2 = B * y;

[C_trn, ac_trn, pr_trn, rc_trn, fa_trn, oa_trn] = Matriz_Confusao(prev_1,
sign_train);

[C_tst, ac_tst, pr_tst, rc_tst, fa_tst, oa_tst] = Matriz_Confusao(prev_2,
sign_test);
```

Resultados - Terminal

```
C_trn =
   106    14      % PV  NF
     8   152      % PF  NV

ac_trn = 0.92143  % Acurácia
pr_trn = 0.92982  % Precisão
rc_trn = 0.88333  % Recall
fa_trn = 0.05     % Falso Alarme
oa_trn = 0.11667  % Omissão de Alarme
```

```
C_tst =
    80     6
    25   169

ac_tst = 0.88929  % Acurácia
pr_tst = 0.7619   % Precisão
rc_tst = 0.93023  % Recall
fa_tst = 0.12887  % Falso Alarme
oa_tst = 0.069767 % Omissão de Alarme
```

Problema 3

O vetor $\bar{y}$ obtido no item anterior, representa os coeficientes da combinação linear das colunas de M_1 que melhor descrevem os dados. O valor $\bar{y_i}$ representa a contribuição da i-ésima variável (x_i) para a identificação do câncer de mama.

Sabendo disso, verificamos as coordenadas com valores mais próximos de 0 e escolhemos, dentro de uma faixa, as colunas (característica) a descartar.

Após alguns testes, percebi que ao consideras apenas as variáveis (x_1, x_3, x_4), conseguimos um resultado que tem um desempenho que se equipara ao modelo atual.

Se diminuíssemos para uma única variável, a mais relevante (x_1), por incrível que pareça, os resultados só seriam um pouco piores, teria aproximadamente $-0,08$ de precisão e acurácia do que quando consideramos essas 3 variáveis. Contudo, a quantidade de omissões subiria ainda mais, por volta de 25% dos pacientes com câncer não seriam identificados.

Percebi que ao diminuir a quantidade de informação, mantendo sempre o que é mais relevante, as taxas de acurácia e precisão diminuem mais rápido do que a taxa de falso alarme cresce.

Retomando o problema, ao considerar as variáveis escolhidas, criamos um novo hiperplano $H_2 \subset \mathbb{R}^4$, tal que

$$H_2 : \beta_0 + \beta_1 x_1 + \beta_3 x_3 + \beta_4 x_4 = 0$$

Bem como obtemos um novo modelo resolvendo para $\bar{y_2}$ a equação:

$$\begin{bmatrix} 1 & \dots & 1 \\ — & {x_1}^T & — \\ — & {x_3}^T & — \\ — & {x_4}^T & — \end{bmatrix} \begin{bmatrix} 1 & | & | & | \\ \vdots & x_1 & x_3 & x_4 \\ 1 & | & | & | \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} 1 & \dots & 1 \\ — & {x_1}^T & — \\ — & {x_3}^T & — \\ — & {x_4}^T & — \end{bmatrix} \begin{bmatrix} | \\ b \\ | \end{bmatrix}$$

$$\implies {(N_1)}^T \, N_1 \, \bar{y_2} = {(N_1)}^T \, b$$

Resolvendo no MATLAB, obtemos

$$\bar{y_2} = \begin{bmatrix} -4.267 \\ -9.502 \\ 21.625 \\ -6.414 \end{bmatrix}$$

Analogamente, definimos N_2, a matriz com as variáveis selecionadas na base dados de tes. Disso, calculamos as previsões desse novo modelo

$$\text{prev}_1 = N_1 * \bar{y_2}$$

$$\text{prev}_2 = N_2 * \bar{y_2}$$

Comparando as previsões e os dados reais, criamos matrizes de confusão. Ao analisar essas matrizes e as métricas advindas delas, constata-se que a adequação do modelo à base de treinamento foi quase tão precisa quanto a anterior, mesmo tendo sido criada usando menos de $\frac{1}{3}$ da informação original.

Outra coisa interessante é que o cenário se inverteu, agora o modelo apresenta uma maior acurácia no teste quando comparado ao treinamento.

Embora as métricas de assertividade (Acurácia, Precisão) estejam relativamente altas, há um aumento considerável nas taxas de "Omissão de Alarme". Sobre a base de treino, essa taxa é 60% maior. Quanto à base de dados teste, uma omissão de alarme é 130% mais provável de acontecer, teríamos que 16% dos pacientes com câncer não seriam notificados ao invés de 6,9%.

Sendo este um modelo para classificar câncer em pacientes, entregar um falso negativo é o que mai se quer evitar. Como a doença é melhor combatida quando identificada em e tratada nos estágios iniciais de seu desenvolvimento, seria muito ruim que alguém que precisasse iniciar um tratamento não o fizesse.

Por fim, constatamos que, de fato, as variáveis não têm a mesma importância para o diagnóstico. Considerando poucas variáveis, é possível treinar um modelo que classifica de forma eficiente os pacientes. Contudo, ao considerar um conjunto maior de informação, algumas métricas são mais favoráveis para a situação. Pensando em mitigar a quantidade de erros graves, acho que o mais adequado é manter o modelo inicial (H_1) que considera todas as variáveis possíveis.

Treinamento

$$C_1^* = \begin{array}{|c|c|}\hline 98 & 22 \\ \hline 10 & 150 \\ \hline \end{array}$$

Acurácia = 0.885 Recall = 0.816 Omissão de Alarme = 0.183

Precisão = 0.907 Falso Alarme = 0.062

Teste

$$C_2^* = \begin{array}{|c|c|}\hline 72 & 14 \\ \hline 11 & 183 \\ \hline \end{array}$$

Acurácia = 0.91 Recall = 0.837 Omissão de Alarme = 0.162

Precisão = 0.867 Falso Alarme = 0.056

3 - Implementação no MATLAB

Questao_3.m - Script

```
% Seleçao das variaveis que contribue mais que 5%
index = abs(y)/sum(abs(y)) > 0.10;
% index = [1,1,0,1,1,0,0,0,0,0,0]
index(1) = 1; % garante que a coluna de 1's permanecerá para o MMQ

% Matriz com apenas as variaveis desejadas
A_2 = A(:,index);
B_2 = B(:,index);

% Novo modelo
y_2 = Gaussian_Elimination_4((A_2' * A_2), (A_2' * sign_train));
% y_2 = [-4.2677, -9.5029, 21.625, -6.4148]

% Novas previsoes
prev_1 = A_2 * y_2;
prev_2 = B_2 * y_2;

% Matrizes de confusao sobre os novos resultados
[C_trn, ac_trn, pr_trn, rc_trn, fa_trn, oa_trn] =  Matriz_Confusao(prev_1,
sign_train);

[C_tst, ac_tst, pr_tst, rc_tst, fa_tst, oa_tst] = Matriz_Confusao(prev_2,
sign_test);
```

Resultados - Terminal

```
C_trn =
    98    22     % PV  NF
    10   150     % PF  NV

ac_trn = 0.88571 % Acurácia
pr_trn = 0.90741 % Precisão
rc_trn = 0.81667 % Recall
fa_trn = 0.0625  % Falso Alarme
oa_trn = 0.18333 % Omissão de Alarme
```

```
C_tst =
    72    14      % PV  NF
    11   183      % PF  NV

ac_tst = 0.91071  % Acurácia
pr_tst = 0.86747  % Precisão
rc_tst = 0.83721  % Recall
fa_tst = 0.056701 % Falso Alarme
oa_tst = 0.16279  % Omissão de Alarme
```